DOI:
10.20532/cit.2019.1004411

Document Version
Peer reviewed version

Diagnostics as a Hybrid Reasoning Processes: from Logic Model to Software design

Matteo Cristani[1], Francesco Olivieri[2], Claudio Tomazzoli[1], Luca Viganò[3], Margherita Zorzi[1]

[1]Department of Computer Science, University of Verona, Italy
[2]Data61, CSIRO, Australia
[3]Department of Informatics, King's College London, UK

Abstract.
Diagnostic tests are used to determine anomalies in complex systems such as organisms or built structures. Once a set of tests is performed, the experts *interpret* their results and make decisions based on them. This process is named *diagnostic reasoning*. It is a process in which a decision is established that uses rules and general knowledge on the tests and the domain.
The artificial intelligence community has focused on devising and automating different methods of diagnosis for medicine and engineering, but, to the best of our knowledge, the decision process in logical terms hasn't yet been investigated thoroughly. The automation of the diagnostic process would be helpful in a number of contexts, in particular when the number of test sets to make decision is too wide to be dealt with manually.
To tackle such challenges, we shall study logical frameworks for diagnostic reasoning, automation methods and their computational properties and technologies implementing these methods.
In this paper, we present the formalization of a hybrid reasoning framework $T^{\perp}$ that hosts *tests* and *deduction rules* on tests, and an algorithm that transforms a $T^{\perp}$ theory into *defeasible logic,* for which an implemented automated deduction technology (called Spindle) exists.
We evaluate the methodology by means of a real-world example related to the Open Web Application Security Project requisites. The full diagnostic process is driven from the definition of the issue to the decision.

Keywords: Tests, Experiments, Hybrid Reasoning, Labelled Logic, Temporal Logic, Defeasible Logic, SPINdle Engine

1 Introduction

Diagnostic reasoning is the process of evaluating the results of operations (questions or practical actions) in order to establish which specific conditions hold on an individual or, generically, a sample. This class of operations are usually called *tests*. A number of scientific fields exploit test-based knowledge acquisition: computer science, engineering, earth sciences, biology, medicine and many others.

A test commonly reveals a *property*, usually in search of anomalies, with a margin of error, and provides information about causes of the anomaly. As a

consequence, by establishing “cause and effect” relationships, tests provide information about possible solutions. Consider medical diagnosis: specific symptoms suggest which tests are to be done, and the results of such tests help the specialists in identifying which disease is currently on and, consequently, which therapies are needed. The steps of this process are *test-driven* and *knowledge-driven* decisions and, therefore, hybrid reasoning processes.

The result of a test is not exact. Tests are prone to errors, for they reveal a property without proving it in a logical way. This is in contrast to what happens in deductive systems, where a reasoning process starts from premises considered true and, through derivation rules, infers consequences. In other words, tests reveal truth on tested conditions in a provisional way.

The diagnostic reasoning processes that we mentioned above are commonly executed on a huge number of data for several specific diagnostic processes, for instance:

1. In information security, when vast system logs and security data are issued and tested.
2. In geology, where the process of testing data for decisions related to anti-earthquake protections are named with the portmanteau word *geognostic* investigations.
3. In medicine, especially in epidemic control population tests, where the number of tests executed and controlled is wide.

In the situations listed above, the decisions to be made are complex and the diagnostic processes *per se* can be time-consuming. It would be therefore worth to assist the specialist (information engineer, earth scientist, medical scientist, etc.) in the process with a computer-assisted diagnostic reasoning system.

In particular, when a diagnostic test is performed, there are numerous configurations of the results that may be complex to treat simultaneously, for a medical doctor or another specialist, and basic automated learning techniques, such as data mining ones, cannot be used for this purpose satisfactorily. Being able to employ reasoning techniques for computer-assisted diagnosis has been one of the main goals of the research efforts on AI in medicine (see [3, 6, 18, 5] for references and specific approaches). In these cases, it would be useful to assist the medical doctor or another specialist in decision making by providing an automated tool able to decide about the logical consistency and the logical consequences of a set of test results, on top of general issues of those tests themselves, including statistical behaviors, temporal relationships and revealing capabilities.

To tackle these challenges, we started a research program that comprises the development of a mechanized reasoning technology, able to decide the

validity of test sets, the assessment of the technology on real-world cases and the comparison with human behaviors in these cases. In this paper, we move the first steps in the research cycle, providing both the definition of a logical framework for the diagnostic reasoning, the T^{L} logic, and the development of an architecture that applies the reasoning process of T^{L} to a real-world case study in information security.

We use the expressiveness of Labelled Modal Logic [14,38,42], with temporal and statistical information added to a basic propositional language. Experiments are modeled in terms of tests viewed as Bayesian classifiers, which reveal one or more properties of a sample.

We define the syntax of formulae and relational rules between labels in T^{L} and sketch ideas about a full deduction systems à la Prawitz, by presenting the deduction rules; however, we do not provide soundness and completeness results as these are beyond the scope of this paper. We propose examples of how T^{L} works and provide technical issues in the construction of the mentioned experimental technology; we also show how to build an architecture to host the developed mechanization of reasoning.

The remainder of the paper is organized as follows. Section 2 discusses some background of this research. Section 3 reviews relevant related literature. Section 4 introduces the logic T^{L}, defining the basic alphabets, the syntax of formulae, test labels (procedures applied during an experiment), and labelled formulae. In particular, Section 4.2 formalizes central notions of the diagnostic-based reasoning, provides a specific analysis of the non-monotonic aspects of the logic itself and focuses also on the relations between tests. In Section 5, we investigate the structure of an architecture for diagnostic reasoning that we further discuss in a real world setting in Section 6. We conclude with Section 7 by summarizing research, discussing some open problems, and sketching future lines of research.

2 Background

In this section we briefly recall some basic notions from statistical information retrieval and learning [9], and we discuss concrete examples of diagnostic procedure.

From a mathematical perspective, a test is naturally interpreted as a statistical classifier, i.e., a function **f** that, fed with an input **a**, is able to predict a probability distribution over a set of classes. Oversimplifying, f assigns to **a** a label **y** that represents the answer (the classification of **a**). This classification is not exact and therefore the answer given by the classifier can be wrong. For example, if **f** encodes the problem “Does **x** enjoy property **P**?”, the answer “Yes” to this question classifies **a** as an element of the set of objects that enjoy

P, and this can be described by an assertion such as **f: P(x)**. There is an implicit epistemic meaning of this assertion, corresponding to the ability of **f** to assert **P(x)** as happens, for instance, in announcement logics or in Agent Communication Languages, where *agents* make assertions, or in the pure epistemic interpretation of the classical modal logic K, where agents know (or believe) assertions. Also in those systems, truth of sentences may not be guaranteed by the assertion, belief or knowledge of the sentences. Someone may assert, believe or know something, but this something might actually be false.

A large taxonomy of probabilistic classifiers has been developed. In this paper, we focus on the simplest type of classifiers, called *(Naive) Bayes (or Bayesian) classifiers*, which exploit some strong statistical assumptions [13]. Bayesian classifiers work well in many complex real-world situations and thus represent the execution of tests in an acceptable way.

Classifiers are prone to error. In this context, errors are described either as false positive results, or false negative results[1]. In the remainder of this paper, we omit the word result(s) whenever it is clear from context; we also speak of true positive and true negative for those answers that coincide with the answers given by a logical formula.

Scientific research in this area aims to reduce errors in Bayesian classifiers, obtaining better methods to derive knowledge from experiments.

The following example both provides a concrete instance of diagnostic reasoning and permits us to introduce the notions of error, and their taxonomy, based on the relation between properties and the revelation of them.

Example 1. *Western-Blot* is a technique used in biology to confirm the existence of antibodies against a particular pathogenic factor. This is determined by the application of the test in a manner that can be considered without false negatives. Western-Blot, however, has a number of false positives. In contrast, the *Elisa test* (or, simply, Elisa) analogously lacks false negatives but it exhibits a larger number of false positives than Western-Blot when applied to the same pathogenic factor.

1 False positives and false negatives are concepts analogous to type I and type II errors in statistical hypothesis testing, where a positive result corresponds to rejecting the null hypothesis and a negative result corresponds to not rejecting the null hypothesis. Roughly speaking, a false positive, commonly called a "false alarm", is a result that indicates that a given condition exists while in fact it does not, whereas a false negative is a test result that indicates that a condition does not hold while in fact it does. In principle, tests can be considered without false negatives when the number of false negative results is irrelevant to the decision process as happens, for instance, for those tests that present 1 case of false negative in 1 million. For the purpose of our logical framework, we can assume that this means that there are no false negatives.

Usually, the sequence of tests depends upon their cost more than their reliability. For instance, Elisa is a cheaper procedure than Western-Blot, and thus Elisa is typically applied before than Western-Blot.
To illustrate this, assume that Elisa answers positively on a given sample. We cannot conclude with certainty that the pathogenic factor is present in the tested organism, due to the high number of false positives exhibited by Elisa. Thus, we apply the Western-Blot test to confirm the validity of Elisa's result. We now derive a negative answer. Since it is assumed that Western-Blot is without false negatives, we can conclude that the pathogenic factor is not present in the organism, against the evidence provided by Elisa.

Example 1 shows a way of deriving truth from tests that is common in those systems. It is straightforward to see that tests with no false negatives that give a negative answer, as well as tests with no false positives that give a positive answer, are always truthful.

3 Related Work

The notion of assisted diagnosis and the usage of intelligent systems in medicine for diagnostic purposes have been a mainstream research topic in artificial intelligence in medicine.

Since the pioneering works of Reiter [6] and Davis [3] these studies have been focusing on two methods: case-based reasoning (see [5] for several recent references to this approach) and statistical methods applied to reasoning (inspired by the original work of Johnson et al. [4]; see [1, 2] for recent investigations).

The nature of errors in tests for diagnostic reasoning has been studied to support the idea that a test has some intrinsic probability of revealing the property it has been devised for. Therefore, the majority of these investigations have focused upon the ideas that a test can be erroneous in making decisions and that, based on a potentially erroneous decision, we have a tree of possible decisions that have a degree of validity, depending strictly on the validity of the starting decision. There is a long stream of investigations based upon fuzzy and probabilistic reasoning methods of computer science applied to medicine, started by Píš et al. in [25] and followed by many other investigations, notably [19] in rheumatology. There have also been some comparative studies, such as [18] and [24]. These methodological studies have given rise to a series of architectural proposals making use of probabilistic methods (see, e.g., [20] and [21]).

4 Focusing on Experimental Knowledge: The Logic $\tau^{\llcorner}$

We introduce logic $\tau^{\llcorner}$ that is devised to perform approximate reasoning on tests. Informally, a (well-formed) formula of $\tau^{\llcorner}$ represents a property of a sample (or individual) that can be revealed, with a margin of error, by a suitable experiment, built out from a sequence of tests. Information about tests is represented by labels, which are metalinguistic logical objects that "adorn" the pure syntactical level of formulae.

4.1 Syntax of $\tau^{\llcorner}$

The *alphabet* of $\tau^{\llcorner}$ is built out of the *variable* symbol **x**, a denumerable set of symbols for *constants* each denoted by lowercase Latin letters (possibly indexed), and a denumerable set of *unary predicates*, denoted by capital Latin letters **P**, **Q** ..., possibly indexed.

Predicates represent properties. When applied to an individual constant, a predicate returns an element of a given domain. Properties are revealed by *tests*, which are not included in the syntax of formulae; rather, we introduce tests in the syntax of *labels* that we give below.

A *ground atomic formula* (ground formula, hereafter) is an atomic formula of the form **P(c)**, where c is a constant. We write **gF** to denote the *set of ground formulae*.

Formulae in $\tau^{\llcorner}$ are built from the set of atomic formulae by means of the usual logical connectives: $\bot$, $\neg$, $\wedge$, $\rightarrow$. Formally, the set **aF** of *well-formed assertion formulae* is the smallest set such that:

(i) $\mathbf{gF} \subseteq \mathbf{aF}$;
(ii) $\bot \in \mathbf{aF}$;
(iii) if $\mathbf{A} \in \mathbf{aF}$ then $\neg A \in \mathbf{aF}$,
(iv) if $\mathbf{A}, \mathbf{B} \in \mathbf{aF}$ then $(\mathbf{A} \wedge \mathbf{B}) \in \mathbf{aF}$, and
(v) if $\mathbf{A}, \mathbf{B} \in \mathbf{aF}$ then $(A \rightarrow B) \in \mathbf{aF}$.

We denote well-formed assertion formulae by **A**, **B**, **C** ..., possibly indexed, and call them formulae or assertions for short.

Basic literals are formed by letters or negations of letters, applied to constants, i.e., **P(c)** and **¬P(c)**. For example, if **Fever** is a predicate and **John** is a constant, then **Fever(John)** is a literal.

Following the tradition of *labelled deduction systems* [14, 10, 11], we extend the syntax above by introducing a class of labels that represent experiments, i.e., instants of time in which tests of properties are performed on a sample, under some environmental conditions. *Labels* are built from a set R of symbols for tests denoted by variables and possibly indexed. Tests in label symbols carry information about the *execution time* (the instant in which the test is performed) and the *experimental condition* (condition, for short),

which is the history of actions performed during the experiment and (possibly) additional information provided/known during the diagnostic process. This reflects the fact that a particular test can be conditioned by a specific situation (like the environment, a medical condition, etc.). For instance, when a geologist conducts a forecast of the position of underground water, among other examinations there is an extraction of a vertical cylinder of ground: if the terrain is very humid, then the stratification of the underground can be different than usual, leading to a change in the forecast itself.

To formalize these ideas, we introduce the set T of symbols for *time instants* t, possibly indexed, and the set A of *experimental conditions* denoted by $\boldsymbol{\varphi}$, possibly indexed. In this paper, which provides a first investigation, we define A simply as the set that contains finite compositional sequences of tests $\boldsymbol{\tau}_1 ... \boldsymbol{\tau}_k$, where we assume that $\boldsymbol{\tau}_{i+1} \in \mathbf{R}$ has been applied after $\boldsymbol{\tau}_i \in R$ on the same sample. Clearly, we can have $\boldsymbol{\varphi} = \varnothing$.

Given a fixed denumerable set Lab_T of labels of the form $\boldsymbol{\tau}^{(t;\varphi)}$, where $\boldsymbol{\tau}$ is a test able to reveal one or more properties**, t** represents a time instant (of a given timeline) and $\boldsymbol{\varphi}$ is the experimental condition. Labels are denoted by l and r, possibly indexed. A test label is a construct that is more expressive than a test symbol: a test label represents a test put into a context, i.e., equipped with additional information such as its time (when it is applies) and the history of the experiment, i.e., the trace of previously applied tests (in the same experiment).

In this paper, we focus on diagnostic reasoning about ground formulae and leave the extension to propositional or first-order logic to future work (see Section 7.

We define labelled formulae as follows. A *labeled (well-formed) formula* is a formula of the form $\boldsymbol{\tau}^{(t;\varphi)}$**: A**, where $A \in gF$. Intuitively, $\boldsymbol{\tau}^{(t;\varphi)}$**: P(c)** denotes the assertion "**$\boldsymbol{\tau}$ reveals P at time t on the sample c, under conditions $\boldsymbol{\varphi}$**". For instance, we can write **Elisa**$^{\mathbf{(Monday;Fever)}}$ **: Ebola(John)** to express that we execute the Elisa test on a sample on Monday, with the patient John having a Fever, to reveal the existence of an infection of Ebola.

Ground facts are ground formulae without labels. We need to introduce one *epistemic* negation to denote the fact that a formula is not revealed by a test, which is conceptually different than stating that a test reveals the negation of a formula. We thus introduce the *negation* ~ that ranges over labelled formulae, in contrast to the logical connective ¬ that we already introduced above. Note that neither $\boldsymbol{\tau}^{(t;\varphi)}$**: A** implies $\boldsymbol{\tau}^{(t;\varphi)}$**: ¬A**, nor $\boldsymbol{\tau}^{(t;\varphi)}$**: ¬A** implies $\boldsymbol{\tau}^{(t;\varphi)}$**: A**.

4.2 Orders and Relation for Tests and Observable Properties

We now discuss the mechanization of experimental reasoning and how to provide a logical foundation of test-based knowledge. In this paper, we mainly focus on test labels and on the reasoning processes performed during a

procedure that aims at extracting experimental knowledge from some resources (typically, a sample).

We can define a partial order between two test labels, both related to temporal information and statistical measures for test performances. We start by defining some temporal orders between labels. We write $t_1 < t_2$ to denote the usual temporal order between time instants, and $\varphi_1 \triangleright \varphi_2$ to denote the order between conditions. We state that $\varphi_1 \triangleright \varphi_2$ indicates that i φ_1 s a prefix of φ_2 .

Following the tradition of labelled deduction systems [14, 10, 11], we define relational formulae by lifting the orders to labels.

Definition 1 (Temporal Relational Formulae).

– $\tau_1^{(t_1;\varphi_1)} \ll \tau_2^{(t_2;\varphi_2)}$ iff $t_1 < t_2$ and $\varphi_1 \triangleright \varphi_2$

– $\tau_1^{(t_1;\varphi_1)} \rightarrow \tau_2^{(t_2;\varphi_2)}$ iff $t_1 < t_2$, $\varphi_2 = \varphi_1 \cdot \tau_1$ and there is no t such that $t_1 < t < t_2$, where $\varphi_1 \cdot \tau_1$ denotes the condition obtained by performing τ_1 after the events described in .

Note that we are modeling the notion of temporal composition of tests. In particular, << represents a general temporal application sequence, whereas $\tau_1 \rightarrow \tau_2$ represents the execution of the test τ_2 immediately after the execution of the test τ_1 . Note also that the above formula requires the introduction of a logic with branching future time (see Section 7).

With a slight abuse of notation, we write $\tau_1 \rightarrow \tau_2$ to denote the test obtained by composing τ_1 and τ_2 ; we treat $\tau_1 \rightarrow \tau_2$ as a symbol in R, and we then use it as a label. We now introduce three orders based on test metrics for elements in Lab_T.

Definition 2 (Metric-based Relational Formulae).

We write

– $\left(\tau_1^{(t_1;\varphi_1)} ¿_a \tau_2^{(t_2;\varphi_2)}\right)$ [A]

if τ_1 at time t_1 and under condition φ_1 is more accurate in revealing A than τ_2 at time t_2 under condition φ_2 .

– $\left(\tau_1^{(t_1;\varphi_1)} ¿_p \tau_2^{(t_2;\varphi_2)}\right)$ [A]

if τ_1 at time t_1 and under condition φ_1 is more precise in revealing A than τ_2 at time t_2 under condition φ_2 .

– $\left(\tau_1^{(t_1;\varphi_1)} ¿_r \tau_2^{(t_2;\varphi_2)}\right)$ [A]

if τ_1 at time t_1 and under condition φ_1 has greater recall in revealing A than τ_2 at time t_2 under condition φ_2 .

The base of empirical reasoning about tests is the deduction of truth on tests that are correct (with no false positives) or complete (with no false

negatives). We introduce the modal operator $\Box_+$ to denote the fact that a test has no false positives, and the modal operator $\Box_-$ to denote that it has no false negatives. The modal operators $\Box_+$ and $\Box_-$ relate to accuracy, precision and recall. We use these terms in the usual meaning they have in machine learning and specifically in the theory of Bayesian classifiers. *Accuracy* is the probabilistic complement of error rate of a test, *precision* is the probabilistic complement of negative error rate (namely the probability of the test giving a correct positive answer), and *recall* is the probabilistic complement of positive error rate (thus the probability of a test giving a correct negative answer). If a test is both correct and complete, then so is the property it reveals.

This can be expressed by means of logical rules. For example, when two tests are differently accurate, and both lack false positives, then they are also ordered in the same way by precision. Analogously, when they lack false negatives, they are also ordered in the same way with respect to recall. We formalize these concepts as follows, where MAR stands for Map Accuracy to Recall, and MAP stands for Map Accuracy to Precision:

$$\frac{\tau_1^{(t,\phi)} <_a \tau_2^{(t,\phi)} \quad \Box_+\tau_1 \quad \Box_+\tau_2}{\tau_1^{(t,\phi)} <_p \tau_2^{(t,\phi)}} MAP \qquad \frac{\tau_1^{(t,\phi)} <_a \tau_2^{(t,\phi)} \quad \Box_-\tau_1 \quad \Box_-\tau_2}{\tau_1^{(t,\phi)} <_r \tau_2^{(t,\phi)}} MAR$$

Interference between $\left(\tau_1^{(t;\varphi)} >_a \tau_2^{(t;\varphi)}\right)$ [A] assertions and $\left(\tau_1^{(t;\varphi)} >_p \tau_2^{(t;\varphi)}\right)$ [A] or $\left(\tau_1^{(t;\varphi)} >_r \tau_2^{(t;\varphi)}\right)$ [A] is managed by means of rules like the following one:

$$\frac{\left(\tau_1^{(t;\varphi)} >_p \tau_2^{(t;\varphi)}\right)[A]\left(\tau_1^{(t;\varphi)} >_r \tau_2^{(t;\varphi)}\right)[A]}{\left(\tau_1^{(t;\varphi)} >_a \tau_2^{(t;\varphi)}\right)[A]} P - R$$

This rule can be reproduced, analogously, for the accuracy as related to recall and to precision.

$$\frac{\left(\tau_1^{(t;\varphi)} >_p \tau_2^{(t;\varphi)}\right)[A]\left(\tau_1^{(t;\varphi)} >_r \tau_2^{(t;\varphi)}\right)[A]}{\left(\tau_1^{(t;\varphi)} >_a \tau_2^{(t;\varphi)}\right)[A]} P - R \frac{\left(\tau_1^{(t;\varphi)} >_p \tau_2^{(t;\varphi)}\right)[A]\left(\tau_1^{(t;\varphi)} >_r \tau_2^{(t;\varphi)}\right)[A]}{\left(\tau_1^{(t;\varphi)} >_a \tau_2^{(t;\varphi)}\right)[A]} A - P$$

The modal interplay between different metrics is an interesting problem from both the proof theoretical viewpoint and the practical one (related to the software design). We leave to future work the implementation of the interplay between different metrics.

In Example 2 we introduce the key idea that we will exploit in the following: the result of a test is measured by the accuracy hypothesis we assume for the test. For instance, when a test is valued 0.8 accurate, we mean that we believe the test result true in 80% of the cases, whilst we think that the test gives a wrong answer in 20% of the cases.

Example 2. Assume that we execute Elisa (Eli) on sample John (J) to test for HIV. We execute the test on Monday (Mon), under the history of no previous test. The test results positive. Now, since Elisa has no false negatives but has false positives (and is not particularly accurate), we execute Western-Blot (WB) on Tuesday (Tue) to confirm/refute Elisa's result. Western-Blot, obviously is executed with the history of Elisa, which does not interfere with it. The test results negative. Now, since Western-Blot has not false negatives, we conclude that the sample is HIV-free.

$$\frac{Eli^{(Mon,\emptyset)} : HIV(J) \quad WB^{(Tue,Eli)} : \neg HIV(J) \quad Mon < Tue \quad \Box\_WB^{(Tue,Eli)} \quad \Box\_Eli^{(Mon,\emptyset)}}{\neg HIV(J)}$$

Western-Blot is more accurate than Elisa, so $WB^{(Tue;Eli)} >_a Eli^{(Mon;\emptyset)}$.

The example shows that the best accuracy of a test τ_i with respect to a test τ_j induces a first, intuitive notion of prevalence: if revealed formulae are contradictory, we trust the more reliable experiment. This will become central in Section 3.3, when we move toward defeasible theories.

It is well known that, when using tests for revealing properties employed in empirical sciences, a given test can interfere with the result of other tests. For instance, certain therapeutic tests (such as the attempt at solving a dangerous potential bacterial infection by the prophylaxis with antibiotics) can make the results of other tests unreliable.

We say that test τ_1 *obfuscates* test τ_2 if performing τ_1 on a sample before τ_2 diminishes τ_2 's ability to reveal a given property.

On the other hand, τ_1 *gifts* a property on a test τ_2 when its application extends the ability of τ_2 to reveal the property itself.

This reasoning is based on the application of tests in sequence, which is the reason why we have introduced an implicit notion of time. We assume that time is discrete, and that tests are executed at a given instant of time. We introduce a notion of absolute time and associate directly temporal instants to test execution only. Partial obfuscation and partial gift can be intuitively described as follows:

– We say that a test τ_1 (for a property A) *a-obfuscates* ($\searrow$a) the test τ_2 of a property B if, when τ_1 is executed before τ_2 , then the accuracy of τ_2 : B is less than it would have been if the test τ_1 on A was not executed.

– We say that a test τ_1 (for a property A) *a-gifts* ($\nearrow$a) a property B if τ_1 : B when, contrary to a-obfuscation, the accuracy of the test for B increases.

We can similarly define *p-obfuscation*, *p-gift*, *r-obfuscation* and *r-gift* referring to obfuscation and gift for precision and recall, instead of accuracy.

More formally, we can provide the following relation, exploiting metric-based relational formulae (Definition 2).

Definition 3 (Obfuscation and Gift).

– $\left(\tau_1^{(t_1;\varphi_1)} \searrow_a \tau_2^{(t_2;\varphi_2)}\right)$ [B] iff $t_1 < t_2$ and $\tau_2^{(t;\varphi)} \square_a \tau_2^{(t_2;\varphi_2)}$ for $t < t_1$ or for φ s.t. $\tau_1 \notin \varphi$

– $\left(\tau_1^{(t_1;\varphi_1)} \nearrow_a \tau_2^{(t_2;\varphi_2)}\right)$ [B] iff $t_1 < t_2$ and $\tau_2^{(t_2;\varphi_2)} \square_a \tau_2^{(t;\varphi)}$ for $t < t_1$ or for φ s.t. $\tau_1 \notin \varphi$

Similar rules for recall and precision can be obtained by replacing relations $\nearrow^a$ and $\searrow_a$ with the counterparts $\nearrow^p$, $\nearrow^r$, $\searrow_p$ and $\searrow_r$.

Total obfuscation and total gift have a specific logical interpretation. A test τ_1 (revealing a property A) *totally obfuscates* another test τ_2 (revealing a property B) if after the execution of τ_1 it is no longer possible to reveal B by means of τ_2 :

$$\frac{\left(\tau_1^{(t_1;\varphi_1)} : A(c)\, \tau_1^{(t_1;\varphi_1)} \uparrow_a \tau_2^{(t_2;\varphi_2)}\right)[B]\, t_1 < t_2}{\tau_1^{(t_1;\varphi_1)} : B(c)}\, totalObf$$

Dually, a test τ_1 (revealing a property A) *totally gifts* another test τ_2 (revealing a property B) if after the execution of τ_1 it is no longer necessary to reveal B, since the information that B holds for the sample is obtained as a side effect of the execution of τ_1. Since B does not require to be revealed but, after the execution of τ_1, it becomes a ground knowledge, we can classify B as a fact.

$$\frac{\left(\tau_1^{(t_1;\varphi_1)} : A(c)\, \tau_1^{(t_1;\varphi_1)} \uparrow_a \tau_2^{(t_2;\varphi_2)}\right)[B]\, t_1 < t_2}{\tau_1^{(t_1;\varphi_1)} : B(c)}\, totalGift$$

From now on, for the sake of simplicity, when writing the total gift and total obfuscation symbols we will omit the “a” symbol so that $\downarrow_a$ and $\uparrow_a$ will simply be $\downarrow$ and $\uparrow$.

Clinical diagnostic is a useful setting to show what kind of hybrid knowledge we are modeling, but it is not the only context in which this knowledge can be found. We discuss here an example related to information

security. In particular, we consider the Open Web Application Security Project (OWASP)[3], Top Ten Most Critical Web Application Security Risks. OWASP's Top Ten is updated regularly and the latest edition includes A_1-Injection, A_2-Broken Authentication, A_3-Sensitive Data Exposure, A_4-XML External Entities (XXE), A_5-Broken Access Control, A_6-Security Misconfiguration, A_7-Cross-Site Scripting (XSS), A_8-Insecure Deserialization, A_9-Using Components with Known Vulnerabilities, A_{10}-Insufficient Logging and Monitoring. These risks are not independent from each other; being exposed to one risk sometimes entails being also exposed to another one in the list. Being exposed to Injection (A_1) means that untrusted data is sent to an interpreter as part of a command or query and this data can trick the interpreter into executing unintended commands or accessing data without proper authorization; this can imply also to be exposed to the risk of Broken Authentication (A_2). In the following example, we interpret vulnerability scanning tools as a test to reveal a given risk of the OWASP Top Ten.

We write $\tau_{(A_i)}^{(t;\varphi)} : A_i$ to say that the security risk A_i is revealed by a test (a suitable scanning tool) $\tau_{(A_i)}$. To classify test/scanning tools, i.e., to measure software performances, we adopt the standard binary classification of algorithm behavior.

Example 3 (Total obfuscation). Let be C a web application. Risk A_9 (Using Components with Known Vulnerabilities) obfuscates risk A_{10} (Insufficient Logging and Monitoring).

$$\frac{\tau_{A_9}^{(Mon,\emptyset)} : A_9(C) \quad \left(\tau_{A_9}^{(Mon,\emptyset)} \downarrow \tau_{A_{10}}^{(Tue,\tau_{A_{10}})}\right)[A_{10}]}{\sim \tau_{A_{10}}^{(Tue,r(\tau_{A_9}))} : A_{10}(C)}$$

Example 4 (Total gift). Let be C a web application. Risk A_1 (Injection) totally gifts risk A_2 (Broken Authentication).

$$\frac{\tau_{A_1}^{(Mon,\emptyset)} : A_1(C) \quad \left(\tau_{A_1}^{(Mon,\emptyset)} \uparrow \tau_{A_2}^{(Tue,\tau_{A_2})}\right)[A_2]}{\tau_{A_1}^{(mon,\emptyset)} : A_1(C)}$$

An interference between tests τ_1 and τ_2 may occur. We write $\tau_1 \perp \tau_2$ if τ_1 and τ_2 are non-interfering, i.e., if they do not obfuscate or gift each other in either direction.

4.3 Defeasible Logic and diagnostic reasoning

One of the most characterising aspects of the experiment based reasoning is the possibility that a property revealed positively by a test is revealed negatively by another test. Generally speaking, we want to devise a method of reasoning that allows us to accommodate contradictory assertions, in a non-monotonic fashion.

In [17,37,39,40,41,43], some of us have investigated the use of Defeasible Logic as a means for managing data coming from external sources and validated by means of data mining methods.

Non-monotonic reasoning accommodates conclusions when dealing with potential conflicts. When derivations may lead to potentially contradictory conclusions, we may typically have two strategies to avoid inconsistencies. In a *credulous* approach, we branch by creating two distinct sets of conclusions: one for each of the contradictory conclusions. Opposite, with a *skeptical* approach, we need a (*preference*) mechanism to establish whether one conclusion is *preferred* to the other one (in literature, this is typically referred to as a *superiority*, or *preference relation;* see [34] for a systematic analysis). If such a mechanism is not able to solve the conflict, no conclusion is derived, unless *exceptions* are given. Exceptions can be seen as particular conditions preventing to draw a specific conclusion[2]. In this paper, we shall not consider credulous settings.

The formalism that we employ here to convert intrinsic non-monotonic aspects of $T^{\perp}$ logic is D*efeasible Logic* (DL), a skeptical non-monotonic reasoning framework that accommodates assertions, priorities and negative exceptions as introduced above.

There are three distinct sources of non-monotonicity when reasoning about tests:

- Two different tests may give out different results on the same sample.
- One test *cannot* be used to conclude a diagnosis, because another test has modified the sample or created a condition that prevents the use of the sample.
- One test *can* be used to conclude diagnosis, because another test has modified the sample or created a condition that allows the use of the sample for concluding on the diagnosis without performing the test at all.

2 Such exceptions are known as *negative exceptions*. In credulous settings, another type of exceptions is possible: *positive exceptions*, whose purpose is to force a particular derivations.

We emphasize these aspects in Section 5, where we introduce a rewriting algorithm that transforms a set of labelled T^L rules into a defeasible theory that can be processed, in turn, by a defeasible engine.

In this perspective, DL can be viewed as a meta-logic: its rules habilitate the expression of diagnostic reasoning in a natural way and, thanks to the rewriting algorithm, a defeasible theory is produced. Once we have produced such a defeasible theory, we can process the theory by means of the reasoning technology SPINdle. In Defeasible Logic [30, 32] we indeed have rules for opposite derivations, although not all concluded. In the situation where rules for opposite literals are activated the logic does not produce any inconsistency but does not draw any conclusion unless a preference (or superiority) relation states that one rule prevails over the other.

A defeasible theory D is defined as a structure (F, R, >), where:

- F is the set of facts, a set of atomic assertions (literals) considered to be always true (e.g., a fact is that "the stove is ON", formally "stove ON"),
- **R** is the set of rules, which in turn contains three finite sets of rules: *strict rules* (denoted by $\rightarrow$), *defeasible rules* (denoted by symbol $\Rightarrow$), and *defeaters* (denoted by symbol ~>).
- > is a binary relation over R, restricted on defeasible rules with opposite conclusions.

A defeasible rule can be defeated by contrary evidence; defeaters are special rules whose only purpose is to defeat defeasible rules by producing contrary evidence. Our framework does not use strict rules or defeaters, but only defeasible rules. The superiority relation establishes that some rules override the conclusion of another one with the opposite conclusion.

Like in [32], we consider only a version of this logic that can be reduced to a propositional theory and does not contain defeaters.

In DL, a *proof P* of length *n* is a finite sequence $P(1), \ldots, P(n)$ of *tagged literals* of the type $\pm\Delta p$ and $\pm\partial p$. The idea is that, at every step of the derivation, a literal is either proven or disproven. The set of positive and negative conclusions is called *extension*. The meanings of tagged literals is as follows

- $+\Delta q$, which means that there is a definite proof for q in D; such a proof uses strict rules and facts only.
- $-\Delta q$ which means that q is definitely refuted in D.
- $+\partial q$ which means that q is defeasibly proven in D.
- ∂q which means that q is defeasibly refuted in D.

Formalisation of the proof tags is out of the scope of the present paper. An idea of how the derivation mechanism works is proposed in the following example.

Example. Let D=({a,b,},R,>) be a defeasible theory such that

R= { r_1: a $\rightarrow$ c

r_2: b $\Rightarrow$ d
r_3: c $\Rightarrow \neg$ d}
>={(r_2, r_3)}.

Then we derive $+\Delta$c via r_1 and $+\partial$d since r_2 is stronger than r_3. Note that it does not matter whether the antecedents of a rule have been proven as strict conclusions: if such literals are used to allow a defeasible rule to fire (as for r_2), the conclusion will be defeasible.

5 An architecture for diagnostic reasoning

5.1 An algorithm for transforming $T^{\perp}$ assertions onto defeasible theories

In this section, we define an algorithm that translates diagnostic results in a DL theory. We employ three kinds of objects: (a) facts, (b) rules and (c) priorities, namely superiority relations that establish which rule prevails when conflicting conclusions arise. We treat $T^{\perp}$ as formed by two layers, the first formed by the defeasible objects and the second formed by meta-rules, establishing how to deal with the rules themselves. Essentially, we consider the facts as known truths, incontrovertible data, such as the results of tests without errors, or direct anamnestic data, for instance the age of a patient. Rules are instead the central part of the logical structure and relate the tests to the diagnosis, providing defeasible derivations.

The relations between tests, both temporal and evaluation ones, including gift and obfuscation, are treated as meta-rules, namely rules providing room for derived priorities. In the application of the translation algorithm we show that the meta-rules can be synchronized, translated into defeasible rules, and then used to make a decision about the meaning of a $T^{\perp}$ theory in linear time. First of all, we introduce the intended meaning of the elements of the system. This architecture of the solution is specified in detail in Section 5.2.

Meta-rules are written with two constraints: time and experimental evaluation. In particular, these rules are transformed into defeasible rules, extended with a temporal label expressing the initial time instant **t** of the (open) interval in which the rule is available to be put before the literals appearing in the rule itself and criteria based on measures, again mapped onto labels in the form $\mathbf{p}^{+}$ or $\mathbf{p}^{-}$ above the derivation operation sign. The transformation algorithm ***SincroCutII*** is introduced below. The algorithm takes as input a set of meta-rules and a set of evaluations of an experiment and transforms them into a defeasible theory by checking the temporal constraints, interference, and modal relations among tests. The result of the algorithm is a defeasible theory. The model of these meta-rules is inspired by studies of one of the authors [8].

We now describe how the algorithm works. It takes in input a finite set of T^L assertions, and gives, as output, a defeasible theory. The first cycle initializes basic data structures, used to host the converted tokens. The second cycle of the algorithm computes the facts in the theory, and therefore determines the base for the subsequent derivations by the Spindle reasoner. The third cycle reads meta-rules and priorities and translates them in the defeasible theory under construction. Notation **r:t** extracts the temporal information of a rule. The procedure ***evalExperiment*** extracts the result of an experiment. The procedure ***createNewRule*** creates a new empty rule.

Algorithm ***SincroCutII***

Input: an ordered set of *Metarules* Θ, current time t and a set of *evaluations* of the experiments Ψ;
Output: a defeasible theory $\mathcal{T} = \langle F, R, P \rangle$ (facts, rules, priorities);
$\Psi' \leftarrow \Psi$
repeat
  $m \leftarrow pop(\Psi')$; $l \leftarrow evalExperiment(m)$; // l *is in the form* $A(l) : C(l) = \tau_i^{(t_i,\phi_i)} : P_k(C)$
  $L \leftarrow push(l, L)$;
until $\Psi' = \emptyset$;
$F \leftarrow \emptyset$; $L' \leftarrow L$;
repeat
  if $l.t_1 \leq$ t **then**
    $f \leftarrow A(l).\phi$; $F \leftarrow push(f, F)$;
    $r' \leftarrow \ ' \Rightarrow' + C(l)$; $R \leftarrow push(r', R)$;
  end if
until $L' = \emptyset$
repeat
  $r \leftarrow pop(\Theta)$;
  if $type(r) = priority$ **then**
    $P \leftarrow push(r, P)$;
  else
    if $r.t_1 \leq$ t **then**
      $\eta \leftarrow \emptyset$; $L' \leftarrow L$;
      repeat
        $l' \leftarrow pop(F')$;
        if $l' \in A(r)$ **then**
          $\eta \leftarrow (A(r) - l')$;
        end if
      until ($L' = \emptyset$ **or** $\eta \neq \emptyset$);
      if $\eta \neq \emptyset$ **then**
        $r' \leftarrow createNewRule()$;
        if $\eta \simeq \downarrow$ **then**
          $r' \leftarrow \ \Rightarrow P_k(C)$; $r'' \leftarrow \ \Rightarrow \sim P_k(C)$;
          $R \leftarrow push(r', R)$; $R \leftarrow push(r'', R)$;
        else if $\eta \simeq \uparrow$ **then**
          $r' \leftarrow \ \Rightarrow P_k(C)$;
          $R \leftarrow push(r', R)$;
        else
          $A(r') \leftarrow \eta$; $C(r') \leftarrow C(r)$; $R \leftarrow push(r', R)$;
        end if
      end if
    end if
  end if
until $\Theta = \emptyset$;
return $\mathcal{T}$;

Observe, moreover, that obfuscation and gift, the relations between rules that are providing room for temporal re-processing of rules themselves, are treated as modals in the second cycle of the algorithm itself.

A certain rule is a candidate for rewriting only if the synchronizer acknowledged that its clock time falls within the validity interval of the rule when the rule has a validity interval explicitly specified, or at the exact instant of the rule if the rule is not tagged so and therefore is considered instantaneous.

The algorithm executes the translation. At this stage of our research, we do not provide proofs of soundness and completeness of the deduction system introduced in Section 2, and, consequently we will not discuss properties of correctness and completeness of the implemented solution. All these investigations are left to future work, along with the discussion of the semantics of the logical framework, and the corresponding canonical models. Being the framework based on DL, clearly the semantics of the first depends upon the semantics of the latter to which the T^L assertions are translated.

What we can prove here is the complexity of the method, which is independent of the issues discussed above (soundness of the T^L deduction rules, semantics of T^L, completeness, canonical models, correctness of the algorithm, completeness of the algorithm). In fact, the algorithm is linear in the number of literals appearing in the T^L set of assertions given as input to the algorithm, since the limit number of cycles that can be executed is the number of literals.

5.2 Architecture of a system implementing T^L transformation

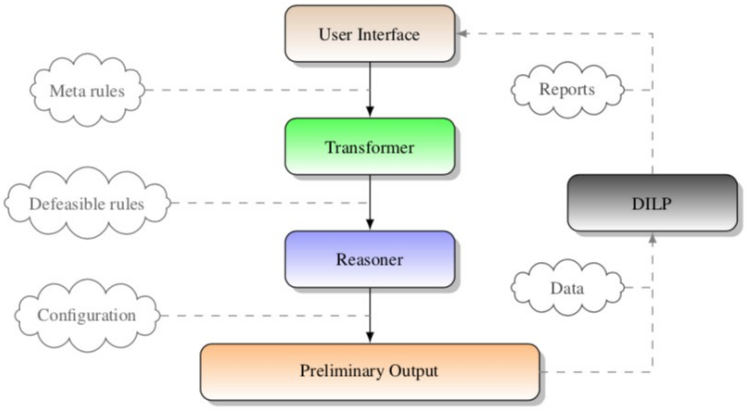

Fig. 1: Logic model of reference architecture

In this section, we briefly introduce an architecture for diagnostic reasoning that is based on four modules, some documented in this paper, some yet to come. The architecture is described in terms of functions of the modules. We introduce here the DILP module, a module used to perform *recommendations* on the rules to

introduce that is based upon the Machine Leaning methods of Inductive Logic Programming, an approach that is also applied to DL.

User Interface: allows the user to input “meta rules” and provides visualization of all data coming from the DILP module;

Transformer: takes as input a set of “meta rules” and gives as output a set of defeasible rules to be used by the Reasoner, according with time given by its internal clock mechanism and an evaluation algorithm;

Reasoner: uses the rules and determines the “should be” conclusions;

Preliminary Output: is responsible for the delivery to the user;

DILP: gathers data and makes analysis delivering summaries and possible rules to be displayed by the User Interface (future extension).

The output of the User Interface is an ordered set of “Meta rules” which are one of the input of the Transformer that runs continuously and at given times uses the algorithm ***SincroCutII*** to produce a set of defeasible rules.

These rules are given to the Reasoner whose output is using the +@ conclusion given the set of rules coming from the Transformer. The Defeasible Logic rule engine used is a Prolog-like engine called SPINdle [33].

At different times different conclusions are possible, due to the work of the Transformer. We now show how the algorithm works by means of a detailed example.

6 Case Study

We use a concrete example to show how our model can fit a real-life scenario. We consider a web application of a bank, located in Italy, which had to be tested against the OWASP Top Ten risks that we listed above.

The analysis is split between two different *contractors*, namely subjects in charge of analyzing a subset of the list. Contractor **α** uses a combination of automated testing and human validation and can cover risk {A1, A3, A4, A5, A10} while contractor **β** performs only automated testing and can cover risk {A2, A4, A6, A7, A8, A9, A10}; **α** is considered to perform tests with a higher accuracy of the ones delivered by **β**. Both execute the tests sequentially one per day and **α** has been engaged after **β** has completed its task. For the sake of space in the rest of this example we will write A_i instead of A_i(BankApp) to describe the ground formulas we are revealing.

There is an obfuscation on A_{10} given a test on A_9 and a gift on A_2 given a test on A_1

$$\frac{\tau_{A_9}^{(t_1,\emptyset)} : A_9 \qquad \left(\tau_{A_9}^{(t_1,\emptyset)} \downarrow \tau_{A_{10}}^{(t_2,\tau_{A_{10}})}\right)[A_{10}]}{\sim \tau_{A_{10}}^{(t_2,r(\tau_{A_9}))} : A_{10}} \qquad \frac{\tau_{A_1}^{(t_1,\emptyset)} : A_1 \qquad \left(\tau_{A_1}^{(t_1,\emptyset)} \uparrow \tau_{A_2}^{(t_2,\tau_{A_2})}\right)[A_2]}{\tau_{A_1}^{(t_1,\emptyset)} : A_2}$$

We also state that **α:A_1 > β:A_1**, **α:A_4 > β:A_4** and **α:A_{10} > β:A_{10}** by knowledge on the accuracy of tests and **α:A_2 >β:A_2** because of the gift specified above. The results are:

$$\{\ \alpha_{A_1}^{(1,\phi_1)} : A_1,\ \alpha_{A_3}^{(2,\phi_2)} : \neg A_3,\ \alpha_{A_4}^{(3,\phi_3)} : A_4,\ \alpha_{A_5}^{(4,\phi_4)} : \neg A_5,\ \alpha_{A_{10}}^{(5,\phi_5)} : A_{10}\ \}$$

$$\{\ \beta_{A_2}^{(6,\phi_6)} : \neg A_2,\ \beta_{A_4}^{(7,\phi_7)} : \neg A_4,\ \beta_{A_6}^{(8,\phi_8)} : A_6,\ \beta_{A_7}^{(9,\phi_9)} : \neg A_7,\ \beta_{A_8}^{(10,\phi_{10})} : A_8,\ \beta_{A_9}^{(11,\phi_{11})} : A_9,$$

$$\beta_{A_{10}}^{(12,\phi_{12})} : \neg A_{10}\}$$

This is therefore the set of meta-rules. Once the Translator has performed Algorithm ***SincroCutII*** we have, at any time after all the tests have been executed (**t > 12**):

$\rightarrow \varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}, \varphi_{11}, \varphi_{12}$

$\rightarrow$

$r\alpha_1 : \Rightarrow A_1$	$r\beta_1 : \Rightarrow \neg A_2$
$r\alpha_2 : \Rightarrow \neg A_3$	$r\beta_2 : \Rightarrow A_4$
$r\alpha_3 : \Rightarrow \neg A_4$	$r\beta_3 : \Rightarrow A_6$
$r\alpha_4 : \Rightarrow \neg A_5$	$r\beta_4 : \Rightarrow \neg A_7$
$r\alpha_5 : \Rightarrow A_{10}$	$r\beta_5 : \Rightarrow A_8$
	$r\beta_6 : \Rightarrow A_9$

$r\ \beta_7 : => \neg A_{10}$

$o_{\beta 1} : => A10$ $g_{\alpha 1} > r_{\beta 1}$

$o_{\beta 2} : => \neg A10$ $r_{\alpha 3} > r_{\beta 2}$

$g_{\alpha 1} => A2$ $r_{\alpha 5} > r_{\beta 7}$

$r_{\alpha 5} > o\beta_2$

Given that theory, the Reasoner concludes $+\partial A_1$, $+\partial A_2$, $+\partial \neg A_3$, $+\partial \neg A_4$, $+\partial \neg A_5$, $+\partial A_6$, $+\partial \neg A_7$, $+\partial A_8$, $+\partial A_9$, $+\partial A_{10}$, as shown in Appendix A.

We can therefore conclude that the application is subject to risks A_1-Injection, A_2-Broken Authentication, A_6-Security Misconfiguration, A_8-Insecure Deserialization, A_9-Using Components with Known Vulnerabilities, A_{10}-Insufficient Logging and Monitoring, but not to A_3-Sensitive Data Exposure, A_4-XML External Entities (XXE), A_5-Broken Access Control, A_7 Cross-Site Scripting (XSS).

7 Discussion and Conclusions

In this paper, we have developed the logic $\tau^{\perp}$ [17], which is able to formalize a form of diagnostic reasoning based both on deduction and on experimental knowledge. We introduced some notions about experiment-based deduction, following a perspective clearly oriented to reasoning mechanization. In comparison with [17], we also focused on the (natural) defeasible aspects of diagnostic knowledge.

To this end, we introduced a rewriting algorithm ***SincroCutII***, that takes as input $\tau^{\perp}$ formulas and transforms them into a defeasible theory by checking temporal, accuracy and interference constraints between tests. The result of the SincroCutII is a defeasible theory. By means of an example from a real-life scenario, we carried out a case study using the defeasible engine SPINdle.

We are currently working in three directions. First, the system $\tau^{\perp}$ can be improved as a (stand-alone) labelled temporal logic framework, and its proof theory seems to be a challenging and interesting task. On the semantic side, we observed that the natural interpretation for $\tau^{\perp}$ is related to some interpretations of the branching time logic UB [7]. The most suitable style is Prawitz' natural deduction [12,35,36]. Following [16, 15], we are developing a labelled, non-monotonic natural deduction system.

Second, the defeasible flavor we pointed out in this paper seems to be the right perspective to move toward a more expressive automatic reasoner. In particular, we aim to extend the deduction system both to include more refined quantitative information about tests and to address more complex diagnostic based deduction, including multi-level defeasible mechanisms.

Finally, there is a strong relation between tests and resources. In a more refined framework, a test could reasonably consume a resource in revealing a property. This reflects what effectively happens in a number of laboratory experiments and we plan to investigate whether this could be captured by means of an approach inspired by linear logic.

References

1. A.E. Lawson and E.S. Daniel: Inferences of clinical diagnostic reasoning and diagnostic error. Journal of Biomedical Informatics, 44(3):402–412, 2011.

2. M. McShane, S. Beale, S. Nirenburg, B. Jarrell, and G. Fantry: Inconsistency as a diagnostic tool in a society of intelligent agents. Artificial Intelligence in Medicine, 55(3):137–148, 2012.

3. R. Davis: Diagnostic reasoning based on structure and behavior. Artificial Intelligence, 24(1-3):347–410, 1984.

4. P.E. Johnson, A.S. Duran, F. Hassebrock, J. Moller, M. Prietula, P.J. Feltovich, and D.B. Swan-son: Expertise and error in diagnostic reasoning. Cognitive Science, 5(3):235–283, 1981.

5. D. McSherry: Conversational case-based reasoning in medical decision making. Artificial Intelligence in Medicine, 52(2):59–66, 2011.

6. R. Reiter: A theory of diagnosis from first principles. Artificial Intelligence, 32(1):57–95, 1987.

7. Caleiro, C., Viganò L., Volpe, M.: A labeled deduction system for the logic UB. Proceedings of the 20th International Symposium on Temporal Representation and Reasoning, TIME, 45– 53 (2013).

8. Cristani, M., Burato, E., Gabrielli, N.: Ontology-Driven Compression of Temporal Series: A Case Study in SCADA Technologies. Proceedings of DEXA Workshop, Turin, Italy, May 2008.

9. Manning, C. D., Raghavan, P., Schütze, H., Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

10. Masini, A., Viganò, L., Zorzi, M.: A Qualitative Modal Representation of Quantum Register Transformations. 38th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2008), 22-23 May 2008, Dallas, Texas, USA, ISMVL 2008: 131-137.

11. Masini, A., Viganò, L., Zorzi, M.: Modal Deduction Systems for Quantum State Transformations, Multiple-Valued Logic and Soft Computing 17(5-6): 475- 519 (2011)

12. Prawitz, D., Natural Deduction: A Proof-Theoretical Study, Almquist and Wiskell, 1965.

13. Rish, I.,An empirical study of the naive Bayes classifier (PDF). IJCAI Workshop on Empirical Methods in AI (2001).

14. Viganò, L.: Labelled Non-Classical Logics. Kluwer Academic Publishers, 2000.

15. Viganò, L., Volpe, M., Zorzi, M.: A branching distributed temporal logic for reasoning about entanglement-free quantum state transformations. Inf. Comput. 255: 311-333 (2017).

16. Viganò, L., Volpe, M., Zorzi, M.: Quantum State Transformations and Branching Distributed Temporal Logic. 21st International Workshop, WoLLIC 2014, Valparaiso, Chile, September 1-4, 2014, Lecture Notes in Computer Science, 8652, 1–19 (2014).

17. Cristani, M., Olivieri, F., Tomazzoli, C., Zorzi, M.: Towards a logical framework for diagnostic reasoning. Smart Innovation, Systems and Technologies 96, KES Conference on Agent and Multi-Agent Systems: Technologies and Applications, KES-AMSTA 2018, pp. 144-155 (2018).

18. J.E.C. Bellamy. Medical diagnosis, diagnostic spaces, and fuzzy systems. Journal of the American Veterinary Medical Association, 210(3):390–396, 1997.

19. M. BelmonteSerrano, C. Sierra, and R.L. de Mantaras. Renoir: An expert system using fuzzy logic for rheumatology diagnosis. International Journal of Intelligent Systems, 9(11):985– 1000, 1994.

20. K. Boegl, K.-P. Adlassnig, Y. Hayashi, T.E. Rothenfluh, and H. Leitich. Knowledge acqui-sition in the fuzzy knowledge representation framework of a medical consultation system. Artificial Intelligence in Medicine, 30(1):1–26, 2004.

21. Q. Liu, F. Jiang, and D. Deng. Design and implement for diagnosis systems of hemorheol-ogy on blood viscosity syndrome based on grc. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2639:413–420, 2003.

22. B. Pandey and R.B. Mishra.Knowledge and intelligent computing system in medicine. Computers in Biology and Medicine, 39(3):215–230, 2009.

23. E.I. Papageorgiou, J.D. Roo, C. Huszka, and D. Colaert. Formalization of treatment guide-lines using fuzzy cognitive maps and semantic web tools. Journal of Biomedical Informatics, 45(1):45–60, 2012.

24. N.H. Phuong and V. Kreinovich. Fuzzy logic and its applications in medicine. International Journal of Medical Informatics, 62(2-3):165–173, 2001.

25. P. Píš and R. Mesiar. Fuzzy model of inexact reasoning in medicine. Computer Methods and Programs in Biomedicine, 30(1):1–8, 1989.

26. G. Rau, K. Becker, R. Kaufmann, and H.J. Zimmermann. Fuzzy logic and control: Principal approach and potential applications in medicine. Artificial Organs, 19(1):105–112, 1995.

27. R. Seising. From vagueness in medical thought to the foundations of fuzzy reasoning in medical diagnosis. Artificial Intelligence in Medicine, 38(3):237–256, 2006.

28. P. Vineis. Methodological insights: Fuzzy sets in medicine. Journal of Epidemiology and Community Health, 62(3):273–278, 2008.

29. A. Yardimci: Soft computing in medicine. Applied Soft Computing Journal, 9(3):1029–1043, 2009.

30. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Pro-gramming, vol. 3. Oxford University Press (1987)

31. G. Antoniou, D. Billington, G. Governatori, M.J. Maher, and A. Rock: A family of defeasible reasoning logics and its implementation. In ECAI 2000, pages 459–463, 2000.

32. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher: Representation results for defeasible logic. ACM Trans. Comput. Log., 2(2):255–287, 2001.

33. Lam, H.P., Governatori, G.: The making of SPINdle. In Paschke, A., Governatori, G., Hall, J., eds.: Proceedings of The International RuleML Symposium on Rule Interchange and Applications (RuleML 2009), Springer (2009) 315–322

34. P.M. Dung, P. Mancarella, and F. Toni: Argumentation-based proof procedures for credulous and skeptical non-monotonic reasoning. Lecture Notes in Computer Science (includ-ing subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2408(PART2):289–310, 2002.

35. Aschieri, A., Zorzi, M.: On natural deduction in classical first-order logic: Curry-Howard correspondence, strong normalization and Herbrand's theorem. Theoretical Computer Science 625: 125-146 (2016).

36. Aschieri, F., Zorzi, M.: Non-determinism, non-termination and the strong normalization of system T. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7941 LNCS, pp. 31-47, 2013.

37. Cristani, M., Olivieri, F., Tomazzoli, C.: Automatic synthesis of best practices for energy consumptions, Proceedings - 2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2016, 7794456, pp. 154-161, 2016.

38. Combi, C., Masini, A., Oliboni, B., Zorzi, M.: A logical framework for XML reference specification. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9262, pp. 258-267, 2015.

39. Cristani, M., Karafili, E., Tomazzoli, C.: Improving energy saving techniques by ambient intelligence scheduling, Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2015

40. Cristani, M., Karafili, E., Tomazzoli, C.: Energy saving by ambient intelligence techniques, Proceedings - 2014 International Conference on Network-Based Information Systems, NBiS 2014.

41. Tomazzoli, C., Cristani, M., Karafili, E., Olivieri, F.: Non-monotonic reasoning rules for energy efficiency, Journal of Ambient Intelligence and Smart Environments, 9 (3), pp. 345-360, 2018.

42. Combi, C., Masini, A., Oliboni, B., Zorzi, M.: A hybrid logic for XML reference constraints. Data knowledge Engineering, 115, pp. 94-115 (2018)

43. Matteo Cristani, Claudio Tomazzoli, Erisa Karafili, Francesco Olivieri: Defeasible Reasoning about Electric Consumptions. AINA 2016: 885-892

A Spindle conclusions for rules of the reference implementation

```
*****************************************
*  SPINdle (version 2.2.4)
*  Copyright (C) 2009-2013 NICTA Ltd.
......
*  java -jar spindle-<version>.jar --app.license
*****************************************************
==========================
== application start!! ==
==========================
Initialize application context - start
.....
====================
......
      +d A1(X) +d
      A10(X) +d
      A2(X) +d
      -A3(X) +d
      -A4(X) +d
      -A5(X) +d
      A6(X) +d
      -A7(X) +d
      A8(X) +d
      A9(X) +d
      Phi1(X) +d
      Phi10(X) +d
      Phi11(X) +d
      Phi12(X) +d
      Phi2(X) +d
      Phi3(X) +d
      Phi4(X) +d
      Phi5(X) +d
      Phi6(X) +d
      Phi7(X) +d
      Phi8(X) +d
      Phi9(X) -d
      -A10(X) -d
      -A2(X) -d
      A4(X)

Calling the shutdown routine...
Terminate application context - start
Terminate application context - end
=====================================
=== Application shutdown completed! ===
=====================================
```